UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/692,320	10/23/2003	Krzysztof J. Cwalina	MS1-1748US	8610

22801 7590 11/16/2007

LEE & HAYES PLLC
421 W RIVERSIDE AVENUE SUITE 500
SPOKANE, WA 99201

EXAMINER	
CHEN, QING	
ART UNIT	**PAPER NUMBER**
2191	
MAIL DATE	**DELIVERY MODE**
11/16/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/692,320	CWALINA ET AL.	
	Examiner	Art Unit	
	Qing Chen	2191	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *11 September 2007*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-49 and 51-59* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-49 and 51-59* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *20070911*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This Office action is in response to the amendment filed on September 11, 2007.

2. **Claims 1-49 and 51-59** are pending.

3. **Claims 1-4, 6-8, 15, 20, 26, 36, 37, 39, 40, 48, 49, 52, 53, and 56** have been amended.

4. **Claim 50** has been cancelled.

5. Applicant has failed to address the objection to the specification due to the use of a trademark. Accordingly, this objection is maintained and further explained below.

6. The objections to Claims 40 and 53 are withdrawn in view of Applicant's amendments to the claims.

7. The 35 U.S.C. § 112, second paragraph, rejections of Claims 2-4, 6-9, 20-22, 26, 36-39, and 49-59 are withdrawn in view of Applicant's amendments to the claims. However, the 35 U.S.C. § 112, second paragraph, rejections of Claims 29-31 and 48 are maintained in view of Applicant's arguments and further explained below.

8. The 35 U.S.C. § 101 rejections of Claims 1-33 and 48-59 are withdrawn in view of Applicant's amendments to the claims and the Office's current policies regarding non-statutory subject matter.

Response to Amendment

Specification

9. The use of trademarks, such as FIREWIRE, has been noted in this application. Trademarks should be capitalized wherever they appear (capitalize each letter OR accompany each trademark with an appropriate designation symbol, *e.g.*, ™ or ®) and be accompanied by the generic terminology (use trademarks as adjectives modifying a descriptive noun, *e.g.*, "the JAVA programming language").

Although the use of trademarks is permissible in patent applications, the proprietary nature of the marks should be respected and every effort made to prevent their use in any manner, which might adversely affect their validity as trademarks.

Claim Rejections - 35 USC § 112

10. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

11. **Claims 29-31 and 48** are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

Claims 29 and 31 recite the limitation "relatively higher/lower." The term "relatively higher/lower" is a relative term, which renders the claims indefinite. The term "relatively higher/lower" is not defined by the claims nor does the specification provide a standard for

ascertaining the requisite degree and one of ordinary skill in the art would not be able to reasonably determine the scope of the invention. In the interest of compact prosecution, the Examiner subsequently does not give any patentable weight to this limitation for the purpose of further examination.

Claim 30 depends on Claim 29 and, therefore, suffers the same deficiency as Claim 29.

Claim 30 recites the limitation "relatively higher." The term "relatively higher" is a relative term, which renders the claim indefinite. The term "relatively higher" is not defined by the claim nor does the specification provide a standard for ascertaining the requisite degree and one of ordinary skill in the art would not be able to reasonably determine the scope of the invention. In the interest of compact prosecution, the Examiner subsequently does not give any patentable weight to this limitation for the purpose of further examination.

Claim 31 recites the limitation "relatively greater." The term "relatively greater" is a relative term, which renders the claim indefinite. The term "relatively greater" is not defined by the claim nor does the specification provide a standard for ascertaining the requisite degree and one of ordinary skill in the art would not be able to reasonably determine the scope of the invention. In the interest of compact prosecution, the Examiner subsequently does not give any patentable weight to this limitation for the purpose of further examination.

Claim 48 recites the limitations "simpler situations" and "complex situations." The terms "simpler" and "complex" are relative terms, which render the claim indefinite. The terms "simpler" and "complex" are not defined by the claim nor does the specification provide a standard for ascertaining the requisite degree and one of ordinary skill in the art would not be able to reasonably determine the scope of the invention. In the interest of compact prosecution, the Examiner subsequently does not give any patentable weight to these limitations for the purpose of further examination.

Claim Rejections - 35 USC § 102

12. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

13. **Claims 15-17, 23-30, 32, 33, and 48** are rejected under 35 U.S.C. 102(e) as being anticipated by **US 7,191,452 (hereinafter Noden)**.

As per **Claim 15**, Noden discloses:

- selecting a core scenario for a feature area *(see Column 5: 28-31, "An external customer may specify the functionality they wish to access via the target/internal application*

through an API request. The external customer further specifies a language for submitting these API requests (i.e., XML).");

- writing at least one code sample for the core scenario *(see Column 5: 32-35, "A manager/programmer for the target application uses a specially designed macro language to create a macro which achieves the specified functionality using the functions exposed by the target/internal application's native API.")*;

- deriving an API for the core scenario responsive to the at least one code sample *(see Column 5: 59-67 to Column 6: 1-4, "Once the desired macros are created, the external customer may utilize the macros in client application(s) and thereby take advantage of the functionality offered by the target application(s). The dispatcher fields new API requests from the network (i.e., via a TCP/IP socket) and may perform some initial validation on the message and extracts environment information from the message to determine which internal application instance this particular API request is intended for ... The message is then queued for execution by a free API engine that may be configured for that particular internal application instance."; Column 6: 61-67, "The macro language is key to how an external API is translated into functions that apply to a particular application. Through a series of instructions comprising one or more macro constructs utilizing internal API calls, SQL statements, and conventional language constructs (i.e., string processing features), the external API macro may be executed against the internal/closed application.")*; and

- realizing the derived API in one or more processor-accessible storage media *(see Column 16: 2-5, "A system for developing intermediate application programming interfaces comprising a processor ...")*.

As per **Claim 16**, the rejection of **Claim 15** is incorporated; and Noden further discloses:

- selecting a plurality of core scenarios for the feature area *(see Column 7: 59-63, "A particular macro may relate to a particular internal application or class of applications (i.e., billing applications) but the technique and the macro language are generic. Thus it is possible that a macro may be coded to serve multiple applications.").*

As per **Claim 17**, the rejection of **Claim 16** is incorporated; and Noden further discloses:

- repeating the writing and the deriving for each core scenario of the plurality of core scenarios that are selected for the feature area *(see Column 5: 59-67 to Column 6: 1-4, "Once the desired macros are created, the external customer may utilize the macros in client application(s) and thereby take advantage of the functionality offered by the target application(s). The dispatcher fields new API requests from the network (i.e., via a TCP/IP socket) and may perform some initial validation on the message and extracts environment information from the message to determine which internal application instance this particular API request is intended for ... The message is then queued for execution by a free API engine that may be configured for that particular internal application instance."; Column 6: 61-67, "The macro language is key to how an external API is translated into functions that apply to a particular application. Through a series of instructions comprising one or more macro constructs utilizing internal API calls, SQL statements, and conventional language constructs (i.e., string processing features), the external API macro may be executed against the internal/closed application.").*

As per **Claim 23**, the rejection of **Claim 15** is incorporated; and Noden further discloses:

- deriving the API to support the at least one code sample written for the core scenario by producing a two-layer API that includes an aggregate component and a plurality of underlying factored types *(see Column 4: 14-17, "An API created and executed, according to the method disclosed herein, can be designed at a high level to provide a complete aggregate function that is self-contained (a `stateless` API)."; Column 7: 48-52, "The macro may then be executed against the internal API using a stub. A stub is a small program routine that substitutes for a longer program. The API stub represents the hook that links the macro to the external program.").*

As per **Claim 24**, the rejection of **Claim 15** is incorporated; and Noden further discloses:

- gleaning one or more language-specific mandates from the at least one code sample *(see Column 6: 55-61, "The macro(s) are processed/executed through a compiler ..." and "The compiler may be designed to process the particular language along with the syntax rules that make up the macro language.")*; and

- incorporating the one or more language-specific mandates into the API *(see Column 6: 61-67, "The macro language is key to how an external API is translated into functions that apply to a particular application. Through a series of instructions comprising one or more macro constructs utilizing internal API calls, SQL statements, and conventional language constructs (i.e., string processing features), the external API macro may be executed against the internal/closed application.").*

As per **Claim 25**, the rejection of **Claim 15** is incorporated; and Noden further discloses:

- encapsulating a particular factored type into an aggregate component that is associated with the core scenario if all members of the particular factored type are exposed by the aggregate component *(see Column 4: 14-17, "An API created and executed, according to the method disclosed herein, can be designed at a high level to provide a complete aggregate function that is self-contained (a `stateless` API)."; Column 7: 48-52, "The macro may then be executed against the internal API using a stub. A stub is a small program routine that substitutes for a longer program. The API stub represents the hook that links the macro to the external program.").*

As per **Claim 26**, the rejection of **Claim 15** is incorporated; and Noden further discloses:

- encapsulating a particular factored type into an aggregate component that is associated with the core scenario if the particular factored type is independently unrelated to other component types *(see Column 4: 14-17, "An API created and executed, according to the method disclosed herein, can be designed at a high level to provide a complete aggregate function that is self-contained (a `stateless` API)."; Column 7: 48-52, "The macro may then be executed against the internal API using a stub. A stub is a small program routine that substitutes for a longer program. The API stub represents the hook that links the macro to the external program.").*

As per **Claim 27**, the rejection of **Claim 15** is incorporated; and Noden further discloses:

- exposing a particular factored type from an aggregate component that is associated with the core scenario if at least one member of the particular factored type is not exposed by the

aggregate component *(see Column 4: 14-17, "An API created and executed, according to the method disclosed herein, can be designed at a high level to provide a complete aggregate function that is self-contained (a `stateless` API)."; Column 7: 48-52, "The macro may then be executed against the internal API using a stub. A stub is a small program routine that substitutes for a longer program. The API stub represents the hook that links the macro to the external program.")*.

As per **Claim 28**, the rejection of **Claim 15** is incorporated; and Noden further discloses:

- exposing a particular factored type from an aggregate component that is associated with the core scenario if the particular factored type can be beneficially used independently of the aggregate component by another component type *(see Column 4: 14-17, "An API created and executed, according to the method disclosed herein, can be designed at a high level to provide a complete aggregate function that is self-contained (a `stateless` API)."; Column 7: 48-52, "The macro may then be executed against the internal API using a stub. A stub is a small program routine that substitutes for a longer program. The API stub represents the hook that links the macro to the external program.")*.

As per **Claim 29**, the rejection of **Claim 15** is incorporated; and Noden further discloses:

- producing a two-layer framework that includes component types targeting a relatively higher level of abstraction and component types targeting a relatively lower level of abstraction *(see Column 4: 14-17, "An API created and executed, according to the method disclosed herein, can be designed at a high level to provide a complete aggregate function that is self-contained (a*

`stateless` API)."; Column 7: 48-52, "The macro may then be executed against the internal API using a stub. A stub is a small program routine that substitutes for a longer program. The API stub represents the hook that links the macro to the external program.").

As per **Claim 30**, the rejection of **Claim 29** is incorporated; and Noden further discloses:

- wherein the component types targeting the relatively higher level of abstraction are directed to core scenarios *(see Column 4: 14-17, "An API created and executed, according to the method disclosed herein, can be designed at a high level to provide a complete aggregate function that is self-contained (a `stateless` API)."; Column 7: 48-52, "The macro may then be executed against the internal API using a stub. A stub is a small program routine that substitutes for a longer program. The API stub represents the hook that links the macro to the external program.").*

As per **Claim 32**, the rejection of **Claim 15** is incorporated; and Noden further discloses:

- deriving the API so as to enable a developer to implement a create-set-call usage pattern for the core scenario *(see Column 5: 59-67 to Column 6: 1-4, "Once the desired macros are created, the external customer may utilize the macros in client application(s) and thereby take advantage of the functionality offered by the target application(s). The dispatcher fields new API requests from the network (i.e., via a TCP/IP socket) and may perform some initial validation on the message and extracts environment information from the message to determine which internal application instance this particular API request is intended for ... The message is then queued for execution by a free API engine that may be configured for that particular*

internal application instance."; Column 6: 61-67, "The macro language is key to how an external API is translated into functions that apply to a particular application. Through a series of instructions comprising one or more macro constructs utilizing internal API calls, SQL statements, and conventional language constructs (i.e., string processing features), the external API macro may be executed against the internal/closed application.").

As per **Claim 33**, the rejection of **Claim 32** is incorporated; and Noden further discloses:

- producing the API with pre-selected parameters that are appropriate for the core scenario *(see Column 5: 59-67 to Column 6: 1-4, "Once the desired macros are created, the external customer may utilize the macros in client application(s) and thereby take advantage of the functionality offered by the target application(s). The dispatcher fields new API requests from the network (i.e., via a TCP/IP socket) and may perform some initial validation on the message and extracts environment information from the message to determine which internal application instance this particular API request is intended for ... The message is then queued for execution by a free API engine that may be configured for that particular internal application instance."; Column 6: 61-67, "The macro language is key to how an external API is translated into functions that apply to a particular application. Through a series of instructions comprising one or more macro constructs utilizing internal API calls, SQL statements, and conventional language constructs (i.e., string processing features), the external API macro may be executed against the internal/closed application.").*

As per **Claim 48**, Noden discloses:

- writing at least one code sample for a scenario *(see Column 5: 32-35, "A manager/programmer for the target application uses a specially designed macro language to create a macro which achieves the specified functionality using the functions exposed by the target/internal application's native API.")*;

- deriving an API for the scenario responsive to the at least one code sample, the API including (i) an aggregate component that is adapted to facilitate implementation of the scenario and (ii) a plurality of factored types that provide underlying functionality for the aggregate component, the API enabling a progression from using the aggregate component in simpler situations to using an increasing portion of the plurality of factored types in increasingly complex situations *(see Column 4: 14-17, "An API created and executed, according to the method disclosed herein, can be designed at a high level to provide a complete aggregate function that is self-contained (a `stateless` API)."; Column 5: 59-67 to Column 6: 1-4, "Once the desired macros are created, the external customer may utilize the macros in client application(s) and thereby take advantage of the functionality offered by the target application(s). The dispatcher fields new API requests from the network (i.e., via a TCP/IP socket) and may perform some initial validation on the message and extracts environment information from the message to determine which internal application instance this particular API request is intended for ... The message is then queued for execution by a free API engine that may be configured for that particular internal application instance."; Column 6: 61-67, "The macro language is key to how an external API is translated into functions that apply to a particular application. Through a series of instructions comprising one or more macro constructs utilizing internal API calls, SQL statements, and conventional language constructs (i.e., string processing features), the external*

API macro may be executed against the internal/closed application."; Column 7: 48-52, "The macro may then be executed against the internal API using a stub. A stub is a small program routine that substitutes for a longer program. The API stub represents the hook that links the macro to the external program."); and

- realizing the derived API in one or more processor-accessible storage media *(see Column 16: 2-5, "A system for developing intermediate application programming interfaces comprising a processor ...").*

Claim Rejections - 35 USC § 103

14. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

15. **Claims 1, 10, 11, 13, 14, 18, and 19** are rejected under 35 U.S.C. 103(a) as being unpatentable over **Noden** in view of **US 5,097,533 (hereinafter Burger)**.

As per **Claim 1**, Noden discloses:

- preparing a core scenario *(see Column 5: 32-35, "A manager/programmer for the target application uses a specially designed macro language to create a macro which achieves the specified functionality using the functions exposed by the target/internal application's native API.")*;

- deriving the API from the core scenario *(see Column 5: 59-67 to Column 6: 1-4, "Once the desired macros are created, the external customer may utilize the macros in client application(s) and thereby take advantage of the functionality offered by the target application(s). The dispatcher fields new API requests from the network (i.e., via a TCP/IP socket) and may perform some initial validation on the message and extracts environment information from the message to determine which internal application instance this particular API request is intended for ... The message is then queued for execution by a free API engine that may be configured for that particular internal application instance."; Column 6: 61-67, "The macro language is key to how an external API is translated into functions that apply to a particular application. Through a series of instructions comprising one or more macro constructs utilizing internal API calls, SQL statements, and conventional language constructs (i.e., string processing features), the external API macro may be executed against the internal/closed application.")*; and

- realizing the derived API in one or more processor-accessible storage media *(see Column 16: 2-5, "A system for developing intermediate application programming interfaces comprising a processor ...")*.

However, Noden does not disclose:

- preparing a plurality of code samples for a core scenario, each respective code sample of the plurality of code samples corresponding to a respective programming language of a plurality of programming languages.

Burger discloses:

- preparing a plurality of code samples for a core scenario, each respective code sample of the plurality of code samples corresponding to a respective programming language of a plurality of programming languages *(see Column 5: 30-36, "In the present invention, however, a plurality of generic APIs 14 are provided to these pre-existing entry points of the DBM APIs 12 and operating system APIs 20 whereby the invention generalizes existing entries to a plurality of applications 16 written in any of a number of predetermined languages.").*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Burger into the teaching of Noden to include preparing a plurality of code samples for a core scenario, each respective code sample of the plurality of code samples corresponding to a respective programming language of a plurality of programming languages. The modification would be obvious because one of ordinary skill in the art would be motivated to reduce development effort, overall maintenance and support involved in providing external entry points to software products from applications written in a wide variety of languages having different interfacing requirements and specifications *(see Burger – Column 3: 19-37).*

As per **Claim 10**, the rejection of **Claim 1** is incorporated; and Noden further discloses:

- deriving the API to support the plurality of code samples that correspond respectively to the plurality of programming languages *(see Column 5: 59-67 to Column 6: 1-4, "Once the desired macros are created, the external customer may utilize the macros in client application(s) and thereby take advantage of the functionality offered by the target application(s). The dispatcher fields new API requests from the network (i.e., via a TCP/IP socket) and may perform*

some initial validation on the message and extracts environment information from the message to determine which internal application instance this particular API request is intended for ... The message is then queued for execution by a free API engine that may be configured for that particular internal application instance."; Column 6: 61-67, "The macro language is key to how an external API is translated into functions that apply to a particular application. Through a series of instructions comprising one or more macro constructs utilizing internal API calls, SQL statements, and conventional language constructs (i.e., string processing features), the external API macro may be executed against the internal/closed application.").

As per **Claim 11**, the rejection of **Claim 1** is incorporated; and Noden further discloses:

- gleaning language-specific mandates from the plurality of code samples *(see Column 6: 55-61, "The macro(s) are processed/executed through a compiler ..." and "The compiler may be designed to process the particular language along with the syntax rules that make up the macro language.")*; and

- incorporating the language-specific mandates into the API *(see Column 6: 61-67, "The macro language is key to how an external API is translated into functions that apply to a particular application. Through a series of instructions comprising one or more macro constructs utilizing internal API calls, SQL statements, and conventional language constructs (i.e., string processing features), the external API macro may be executed against the internal/closed application.").*

As per **Claim 13**, the rejection of **Claim 1** is incorporated; and Noden further discloses:

- gleaning commonalities from the plurality of code samples *(see Column 6: 55-61, "The macro(s) are processed/executed through a compiler ..." and "The compiler may be designed to process the particular language along with the syntax rules that make up the macro language.")*; and

- incorporating the commonalities into the API *(see Column 6: 61-67, "The macro language is key to how an external API is translated into functions that apply to a particular application. Through a series of instructions comprising one or more macro constructs utilizing internal API calls, SQL statements, and conventional language constructs (i.e., string processing features), the external API macro may be executed against the internal/closed application.")*.

As per **Claim 14**, the rejection of **Claim 1** is incorporated; and Noden further discloses:

- deriving the API to have an aggregate component that ties a plurality of lower-level factored types together to support the core scenario *(see Column 4: 14-17, "An API created and executed, according to the method disclosed herein, can be designed at a high level to provide a complete aggregate function that is self-contained (a `stateless` API)."; Column 7: 48-52, "The macro may then be executed against the internal API using a stub. A stub is a small program routine that substitutes for a longer program. The API stub represents the hook that links the macro to the external program.")*.

As per **Claim 18**, the rejection of **Claim 15** is incorporated; however, Noden does not disclose:

- writing a plurality of code samples for the core scenario, each respective code sample of the plurality of code samples corresponding to a respective programming language of a plurality of programming languages.

Burger discloses:

- writing a plurality of code samples for the core scenario, each respective code sample of the plurality of code samples corresponding to a respective programming language of a plurality of programming languages *(see Column 5: 30-36, "In the present invention, however, a plurality of generic APIs 14 are provided to these pre-existing entry points of the DBM APIs 12 and operating system APIs 20 whereby the invention generalizes existing entries to a plurality of applications 16 written in any of a number of predetermined languages.")*.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Burger into the teaching of Noden to include writing a plurality of code samples for the core scenario, each respective code sample of the plurality of code samples corresponding to a respective programming language of a plurality of programming languages. The modification would be obvious because one of ordinary skill in the art would be motivated to reduce development effort, overall maintenance and support involved in providing external entry points to software products from applications written in a wide variety of languages having different interfacing requirements and specifications *(see Burger – Column 3: 19-37)*.

As per **Claim 19**, the rejection of **Claim 18** is incorporated; and Noden further discloses:

- deriving the API for the core scenario responsive to the plurality of code samples *(see Column 5: 59-67 to Column 6: 1-4, "Once the desired macros are created, the external customer may utilize the macros in client application(s) and thereby take advantage of the functionality offered by the target application(s). The dispatcher fields new API requests from the network (i.e., via a TCP/IP socket) and may perform some initial validation on the message and extracts environment information from the message to determine which internal application instance this particular API request is intended for ... The message is then queued for execution by a free API engine that may be configured for that particular internal application instance."; Column 6: 61-67, "The macro language is key to how an external API is translated into functions that apply to a particular application. Through a series of instructions comprising one or more macro constructs utilizing internal API calls, SQL statements, and conventional language constructs (i.e., string processing features), the external API macro may be executed against the internal/closed application.").*

16. **Claims 2-4** are rejected under 35 U.S.C. 103(a) as being unpatentable over **Noden** in view of **Burger** as applied to Claim 1 above, and further in view of **US 6,006,279 (hereinafter Hayes)**.

As per **Claim 2**, the rejection of **Claim 1** is incorporated; however, Noden and Burger do not disclose:

- determining, by an API designer, if the derived API is more complex than desired.

Hayes discloses:

- determining, by an API designer, if the derived API is more complex than desired *(see Column 1: 35-37, "... the Photoshop native plug-in API is defined by Adobe developers." and 40-42, "Plug-in API's have become more complex as the functionality provided to client applications through plug-in modules has become more sophisticated.").*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Hayes into the teaching of Noden to include determining, by an API designer, if the derived API is more complex than desired. The modification would be obvious because one of ordinary skill in the art would be motivated to provide a simple API *(see Hayes – Column 1: 60-63).*

As per **Claim 3**, the rejection of **Claim 2** is incorporated; however, Noden and Burger do not disclose:

- if the derived API is determined to be more complex than desired, then refining, by the API designer, the derived API to produce a refined API.

Hayes discloses:

- if the derived API is determined to be more complex than desired, then refining, by the API designer, the derived API to produce a refined API *(see Column 1: 66-67 through Column 2: 1-3, "It is therefore an object of the invention to provide a host framework which allows a client application to utilize a single simple interface in order to access plug-in modules conforming to any of several past and future native plug-in API's.").*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Hayes into the teaching of Noden to include if

the derived API is determined to be more complex than desired, then refining, by the API designer, the derived API to produce a refined API. The modification would be obvious because one of ordinary skill in the art would be motivated to provide a simple API *(see Hayes – Column 1: 60-63)*.

As per **Claim 4**, the rejection of **Claim 3** is incorporated; however, Noden and Burger do not disclose:

- determining, by the API designer, if the refined API is more complex than desired.

Hayes discloses:

- determining, by the API designer, if the refined API is more complex than desired *(see Column 1: 35-37, "... the Photoshop native plug-in API is defined by Adobe developers." and 40-42, "Plug-in API's have become more complex as the functionality provided to client applications through plug-in modules has become more sophisticated.")*.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Hayes into the teaching of Noden to include determining, by the API designer, if the refined API is more complex than desired. The modification would be obvious because one of ordinary skill in the art would be motivated to provide a simple API *(see Hayes – Column 1: 60-63)*.

17. **Claims 5-9, 12, and 47** are rejected under 35 U.S.C. 103(a) as being unpatentable over **Noden** in view of **Burger** as applied to Claims 1 above, and further in view of **US 5,495,571 (hereinafter Corrie)**.

As per **Claim 5**, the rejection of **Claim 1** is incorporated; however, Noden and Burger do not disclose:

- performing one or more usability studies on the API utilizing a plurality of developers.

Corrie discloses:

- performing one or more usability studies on the API utilizing a plurality of developers *(see Column 2: 4-8, "When testing all of the functions in the application programming interface to Microsoft Windows, for example, programmers must write modules of test code for approximately 1,000 functions, each function having on an average of 3-4 parameters.").*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Corrie into the teaching of Noden to include performing one or more usability studies on the API utilizing a plurality of developers. The modification would be obvious because one of ordinary skill in the art would be motivated to save money by preventing expensive product updates to correct errors, and the high product quality ensured by the thorough testing would enhance a software product's reputation and marketability *(see Corrie – Column 2: 22-26).*

As per **Claim 6**, the rejection of **Claim 5** is incorporated; however, Noden and Burger do not disclose:

- performing the one or more usability studies on the API utilizing the plurality of developers wherein the plurality of developers are competent with the plurality of programming languages.

Corrie discloses:

- performing the one or more usability studies on the API utilizing the plurality of developers wherein the plurality of developers are competent with the plurality of programming languages *(see Column 2: 4-8, "When testing all of the functions in the application programming interface to Microsoft Windows, for example, programmers must write modules of test code for approximately 1,000 functions, each function having on an average of 3-4 parameters.").*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Corrie into the teaching of Noden to include performing the one or more usability studies on the API utilizing the plurality of developers wherein the plurality of developers are competent with the plurality of programming languages. The modification would be obvious because one of ordinary skill in the art would be motivated to save money by preventing expensive product updates to correct errors, and the high product quality ensured by the thorough testing would enhance a software product's reputation and marketability *(see Corrie – Column 2: 22-26).*

As per **Claim 7**, the rejection of **Claim 5** is incorporated; however, Noden and Burger do not disclose:

- ascertaining whether the plurality of developers are able to use the API without problems.

Corrie discloses:

- ascertaining whether the plurality of developers are able to use the API without problems *(see Column 1: 52-57, "Programmers perform parametric testing of functions to test how functions react to valid and invalid parameters.")*.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Corrie into the teaching of Noden to include ascertaining whether the plurality of developers are able to use the API without problems. The modification would be obvious because one of ordinary skill in the art would be motivated to save money by preventing expensive product updates to correct errors, and the high product quality ensured by the thorough testing would enhance a software product's reputation and marketability *(see Corrie – Column 2: 22-26)*.

As per **Claim 8**, the rejection of **Claim 7** is incorporated; however, Noden and Burger do not disclose:

- if the plurality of developers are not ascertained to be able to use the API without problems, then revising the API.

Corrie discloses:

- if the plurality of developers are not ascertained to be able to use the API without problems, then revising the API *(see Column 1: 52-57, "To perform parametric testing of a function in prior art systems, programmers write modules of test code to pass valid and invalid parameters to the function, and then observe the behavior of the function."; Column 2: 52-55, "The test utility coordinates the parametric testing of each function in the functional*

programming interface. The test utility reads the script file and generates a plurality of unique test cases.").

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Corrie into the teaching of Noden to include if the plurality of developers are not ascertained to be able to use the API without problems, then revising the API. The modification would be obvious because one of ordinary skill in the art would be motivated to save money by preventing expensive product updates to correct errors, and the high product quality ensured by the thorough testing would enhance a software product's reputation and marketability *(see Corrie – Column 2: 22-26).*

As per **Claim 9**, the rejection of **Claim 8** is incorporated; however, Noden and Burger do not disclose:

- revising the API based on at least one lesson from the one or more usability studies.

Corrie discloses:

- revising the API based on at least one lesson from the one or more usability studies *(see Column 2: 52-55, "The test utility coordinates the parametric testing of each function in the functional programming interface. The test utility reads the script file and generates a plurality of unique test cases.").*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Corrie into the teaching of Noden to include revising the API based on at least one lesson from the one or more usability studies. The modification would be obvious because one of ordinary skill in the art would be motivated to

save money by preventing expensive product updates to correct errors, and the high product quality ensured by the thorough testing would enhance a software product's reputation and marketability *(see Corrie – Column 2: 22-26)*.

As per **Claim 12**, the rejection of **Claim 1** is incorporated; however, Noden and Burger do not disclose:

- gleaning language-inspired developer expectations from the plurality of code samples; and
- incorporating the language-inspired developer expectations into the API.

Corrie discloses:

- gleaning language-inspired developer expectations from the plurality of code samples *(see Column 1: 52-57, "To perform parametric testing of a function in prior art systems, programmers write modules of test code to pass valid and invalid parameters to the function, and then observe the behavior of the function.")*; and
- incorporating the language-inspired developer expectations into the API *(see Column 2: 52-55, "The test utility coordinates the parametric testing of each function in the functional programming interface. The test utility reads the script file and generates a plurality of unique test cases.")*.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Corrie into the teaching of Noden to include gleaning language-inspired developer expectations from the plurality of code samples; and incorporating the language-inspired developer expectations into the API. The modification

would be obvious because one of ordinary skill in the art would be motivated to save money by preventing expensive product updates to correct errors, and the high product quality ensured by the thorough testing would enhance a software product's reputation and marketability *(see Corrie – Column 2: 22-26).*

As per **Claim 47**, Noden discloses:

- deriving the API for the core scenario *(see Column 5: 59-67 to Column 6: 1-4, "Once the desired macros are created, the external customer may utilize the macros in client application(s) and thereby take advantage of the functionality offered by the target application(s). The dispatcher fields new API requests from the network (i.e., via a TCP/IP socket) and may perform some initial validation on the message and extracts environment information from the message to determine which internal application instance this particular API request is intended for ... The message is then queued for execution by a free API engine that may be configured for that particular internal application instance."; Column 6: 61-67, "The macro language is key to how an external API is translated into functions that apply to a particular application. Through a series of instructions comprising one or more macro constructs utilizing internal API calls, SQL statements, and conventional language constructs (i.e., string processing features), the external API macro may be executed against the internal/closed application.").*

However, Noden does not disclose:

- preparing a plurality of code samples for a core scenario, each respective code sample of the plurality of code samples corresponding to a respective programming language of a plurality of programming languages;

- performing one or more usability studies on the API utilizing a plurality of developers; and

- revising the API based on the one or more usability studies.

Burger discloses:

- preparing a plurality of code samples for a core scenario, each respective code sample of the plurality of code samples corresponding to a respective programming language of a plurality of programming languages *(see Column 5: 30-36, "In the present invention, however, a plurality of generic APIs 14 are provided to these pre-existing entry points of the DBM APIs 12 and operating system APIs 20 whereby the invention generalizes existing entries to a plurality of applications 16 written in any of a number of predetermined languages.").*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Burger into the teaching of Noden to include preparing a plurality of code samples for a core scenario, each respective code sample of the plurality of code samples corresponding to a respective programming language of a plurality of programming languages. The modification would be obvious because one of ordinary skill in the art would be motivated to reduce development effort, overall maintenance and support involved in providing external entry points to software products from applications written in a wide variety of languages having different interfacing requirements and specifications *(see Burger – Column 3: 19-37).*

Corrie discloses:

- performing one or more usability studies on the API utilizing a plurality of developers *(see Column 2: 4-8, "When testing all of the functions in the application programming interface to Microsoft Windows, for example, programmers must write modules of test code for approximately 1,000 functions, each function having on an average of 3-4 parameters.")*; and
- revising the API based on the one or more usability studies *(see Column 2: 52-55, "The test utility coordinates the parametric testing of each function in the functional programming interface. The test utility reads the script file and generates a plurality of unique test cases.")*.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Corrie into the teaching of Noden to include performing one or more usability studies on the API utilizing a plurality of developers; and revising the API based on the one or more usability studies. The modification would be obvious because one of ordinary skill in the art would be motivated to save money by preventing expensive product updates to correct errors, and the high product quality ensured by the thorough testing would enhance a software product's reputation and marketability *(see Corrie – Column 2: 22-26)*.

18. **Claims 20-22, 31, 34, and 37-46** are rejected under 35 U.S.C. 103(a) as being unpatentable over **Noden** in view of **Corrie**.

As per **Claim 20**, the rejection of **Claim 15** is incorporated; however, Noden does not disclose:

- performing one or more usability studies on the API utilizing a plurality of developers;

- ascertaining whether the plurality of developers are able to use the API without problems; and

- if the plurality of developers are not ascertained to be able to use the API without problems, then revising the API.

Corrie discloses:

- performing one or more usability studies on the API utilizing a plurality of developers *(see Column 2: 4-8, "When testing all of the functions in the application programming interface to Microsoft Windows, for example, programmers must write modules of test code for approximately 1,000 functions, each function having on an average of 3-4 parameters.")*;

- ascertaining whether the plurality of developers are able to use the API without problems *(see Column 1: 52-57, "Programmers perform parametric testing of functions to test how functions react to valid and invalid parameters.")*; and

- if the plurality of developers are not ascertained to be able to use the API without problems, then revising the API *(see Column 1: 52-57, "To perform parametric testing of a function in prior art systems, programmers write modules of test code to pass valid and invalid parameters to the function, and then observe the behavior of the function."; Column 2: 52-55, "The test utility coordinates the parametric testing of each function in the functional*

programming interface. The test utility reads the script file and generates a plurality of unique test cases.").

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Corrie into the teaching of Noden to include performing one or more usability studies on the API utilizing a plurality of developers; ascertaining whether the plurality of developers are able to use the API without problems; and if the plurality of developers are not ascertained to be able to use the API without problems, then revising the API. The modification would be obvious because one of ordinary skill in the art would be motivated to save money by preventing expensive product updates to correct errors, and the high product quality ensured by the thorough testing would enhance a software product's reputation and marketability *(see Corrie – Column 2: 22-26).*

As per **Claim 21**, the rejection of **Claim 20** is incorporated; however, Noden does not disclose:

- revising the API based on at least one lesson from the one or more usability studies to produce a revised API.

Corrie discloses:

- revising the API based on at least one lesson from the one or more usability studies to produce a revised API *(see Column 2: 52-55, "The test utility coordinates the parametric testing of each function in the functional programming interface. The test utility reads the script file and generates a plurality of unique test cases.").*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Corrie into the teaching of Noden to include revising the API based on at least one lesson from the one or more usability studies to produce a revised API. The modification would be obvious because one of ordinary skill in the art would be motivated to save money by preventing expensive product updates to correct errors, and the high product quality ensured by the thorough testing would enhance a software product's reputation and marketability *(see Corrie – Column 2: 22-26)*.

As per **Claim 22**, the rejection of **Claim 21** is incorporated; however, Noden does not disclose:

- repeating the performing and the ascertaining with respect to the revised API.

Corrie discloses:

- repeating the performing and the ascertaining with respect to the revised API *(see Column 1: 52-57, "Programmers perform parametric testing of functions to test how functions react to valid and invalid parameters."; Column 2: 4-8, "When testing all of the functions in the application programming interface to Microsoft Windows, for example, programmers must write modules of test code for approximately 1,000 functions, each function having on an average of 3-4 parameters.")*.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Corrie into the teaching of Noden to include repeating the performing and the ascertaining with respect to the revised API. The modification would be obvious because one of ordinary skill in the art would be motivated to save money by

preventing expensive product updates to correct errors, and the high product quality ensured by the thorough testing would enhance a software product's reputation and marketability *(see Corrie – Column 2: 22-26)*.

As per **Claim 31**, the rejection of **Claim 29** is incorporated; however, Noden does not disclose:

- wherein the component types targeting the relatively lower level of abstraction provide a relatively greater amount of control to developers as compared to the component types targeting the relatively higher level of abstraction.

Corrie discloses:

- wherein the component types targeting the relatively lower level of abstraction provide a relatively greater amount of control to developers as compared to the component types targeting the relatively higher level of abstraction *(see Column 1: 52-57, "To perform parametric testing of a function in prior art systems, programmers write modules of test code to pass valid and invalid parameters to the function, and then observe the behavior of the function.")*.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Corrie into the teaching of Noden to include wherein the component types targeting the relatively lower level of abstraction provide a relatively greater amount of control to developers as compared to the component types targeting the relatively higher level of abstraction. The modification would be obvious because one of ordinary skill in the art would be motivated to save money by preventing expensive product

updates to correct errors, and the high product quality ensured by the thorough testing would enhance a software product's reputation and marketability *(see Corrie –Column 2: 22-26)*.

As per **Claim 34**, Noden discloses:

- deriving an API for a scenario responsive to at least one code sample written with regard to the scenario *(see Column 5: 59-67 to Column 6: 1-4, "Once the desired macros are created, the external customer may utilize the macros in client application(s) and thereby take advantage of the functionality offered by the target application(s). The dispatcher fields new API requests from the network (i.e., via a TCP/IP socket) and may perform some initial validation on the message and extracts environment information from the message to determine which internal application instance this particular API request is intended for ... The message is then queued for execution by a free API engine that may be configured for that particular internal application instance."; Column 6: 61-67, "The macro language is key to how an external API is translated into functions that apply to a particular application. Through a series of instructions comprising one or more macro constructs utilizing internal API calls, SQL statements, and conventional language constructs (i.e., string processing features), the external API macro may be executed against the internal/closed application.")*.

However, Noden does not disclose:

- performing one or more usability studies on the API utilizing a plurality of developers; and

- revising the API based on the one or more usability studies.

Corrie discloses:

- performing one or more usability studies on the API utilizing a plurality of developers *(see Column 2: 4-8, "When testing all of the functions in the application programming interface to Microsoft Windows, for example, programmers must write modules of test code for approximately 1,000 functions, each function having on an average of 3-4 parameters.")*; and

- revising the API based on the one or more usability studies *(see Column 2: 52-55, "The test utility coordinates the parametric testing of each function in the functional programming interface. The test utility reads the script file and generates a plurality of unique test cases.")*.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Corrie into the teaching of Noden to include performing one or more usability studies on the API utilizing a plurality of developers; and revising the API based on the one or more usability studies. The modification would be obvious because one of ordinary skill in the art would be motivated to save money by preventing expensive product updates to correct errors, and the high product quality ensured by the thorough testing would enhance a software product's reputation and marketability *(see Corrie – Column 2: 22-26)*.

As per **Claim 37**, the rejection of **Claim 34** is incorporated; however, Noden does not disclose:

- ascertaining whether the plurality of developers are able to use the API without problems; and

- when the plurality of developers are not ascertained to be able to use the API without problems, then implementing the revising; wherein the revising comprises: revising the API based on at least one lesson from the one or more usability studies to produce a revised API.

Corrie discloses:

- ascertaining whether the plurality of developers are able to use the API without problems *(see Column 1: 52-57, "Programmers perform parametric testing of functions to test how functions react to valid and invalid parameters.")*; and

- when the plurality of developers are not ascertained to be able to use the API without problems, then implementing the revising; wherein the revising comprises: revising the API based on at least one lesson from the one or more usability studies to produce a revised API *(see Column 1: 52-57, "To perform parametric testing of a function in prior art systems, programmers write modules of test code to pass valid and invalid parameters to the function, and then observe the behavior of the function."; Column 2: 52-55, "The test utility coordinates the parametric testing of each function in the functional programming interface. The test utility reads the script file and generates a plurality of unique test cases.")*.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Corrie into the teaching of Noden to include ascertaining whether the plurality of developers are able to use the API without problems; and when the plurality of developers are not ascertained to be able to use the API without problems, then implementing the revising; wherein the revising comprises: revising the API based on at least one lesson from the one or more usability studies to produce a revised API. The modification would be obvious because one of ordinary skill in the art would be motivated to

save money by preventing expensive product updates to correct errors, and the high product quality ensured by the thorough testing would enhance a software product's reputation and marketability *(see Corrie – Column 2: 22-26)*.

As per **Claim 38**, the rejection of **Claim 37** is incorporated; however, Noden does not disclose:

- repeating at least the performing and the ascertaining with respect to the revised API.

Corrie discloses:

- repeating at least the performing and the ascertaining with respect to the revised API *(see Column 1: 52-57, "Programmers perform parametric testing of functions to test how functions react to valid and invalid parameters."; Column 2: 4-8, "When testing all of the functions in the application programming interface to Microsoft Windows, for example, programmers must write modules of test code for approximately 1,000 functions, each function having on an average of 3-4 parameters.")*.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Corrie into the teaching of Noden to include repeating at least the performing and the ascertaining with respect to the revised API. The modification would be obvious because one of ordinary skill in the art would be motivated to save money by preventing expensive product updates to correct errors, and the high product quality ensured by the thorough testing would enhance a software product's reputation and marketability *(see Corrie – Column 2: 22-26)*.

As per **Claim 39**, the rejection of **Claim 37** is incorporated; however, Noden do not disclose:

- ascertaining whether the plurality of developers are able to use the API without problems with regard to a desired level of usability for at least one targeted developer group, wherein the desired level of usability includes considerations with respect to (i) frequent and/or extensive reference to detailed API documentation by the plurality of developers, (ii) a failure of a majority of the plurality of developers to implement the scenario, and (iii) whether the plurality of developers take an approach that is different from what is expected by an API designer.

Corrie discloses:

- ascertaining whether the plurality of developers are able to use the API without problems with regard to a desired level of usability for at least one targeted developer group, wherein the desired level of usability includes considerations with respect to (i) frequent and/or extensive reference to detailed API documentation by the plurality of developers, (ii) a failure of a majority of the plurality of developers to implement the scenario, and (iii) whether the plurality of developers take an approach that is different from what is expected by an API designer *(see Column 1: 52-57, "Programmers perform parametric testing of functions to test how functions react to valid and invalid parameters."; Column 2: 33-45, "A script file, a function library, and a test utility are all used in this testing process. The script file contains prototype information for each function to be tested and a case specification for each parameter of the function.")*.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Corrie into the teaching of Noden to include ascertaining whether the plurality of developers are able to use the API without problems with

regard to a desired level of usability for at least one targeted developer group, wherein the desired level of usability includes considerations with respect to (i) frequent and/or extensive reference to detailed API documentation by the plurality of developers, (ii) a failure of a majority of the plurality of developers to implement the scenario, and (iii) whether the plurality of developers take an approach that is different from what is expected by an API designer. The modification would be obvious because one of ordinary skill in the art would be motivated to save money by preventing expensive product updates to correct errors, and the high product quality ensured by the thorough testing would enhance a software product's reputation and marketability *(see Corrie – Column 2: 22-26)*.

As per **Claim 40**, the rejection of **Claim 34** is incorporated; and Noden further discloses:

- selecting a plurality of core scenarios for a feature area *(see Column 7: 59-63, "A particular macro may relate to a particular internal application or class of applications (i.e., billing applications) but the technique and the macro language are generic. Thus it is possible that a macro may be coded to serve multiple applications.")*; and

- repeating the deriving for each core scenario of the plurality of core scenarios *(see Column 5: 59-67 to Column 6: 1-4, "Once the desired macros are created, the external customer may utilize the macros in client application(s) and thereby take advantage of the functionality offered by the target application(s). The dispatcher fields new API requests from the network (i.e., via a TCP/IP socket) and may perform some initial validation on the message and extracts environment information from the message to determine which internal application instance this particular API request is intended for ... The message is then queued for execution by a free API*

engine that may be configured for that particular internal application instance."; Column 6: 61-67, "The macro language is key to how an external API is translated into functions that apply to a particular application. Through a series of instructions comprising one or more macro constructs utilizing internal API calls, SQL statements, and conventional language constructs (i.e., string processing features), the external API macro may be executed against the internal/closed application.").

However, Noden does not disclose:

- repeating the performing and the revising for each core scenario of the plurality of core scenarios.

Corrie discloses:

- repeating the performing and the revising for each core scenario of the plurality of core scenarios *(see Column 2: 4-8, "When testing all of the functions in the application programming interface to Microsoft Windows, for example, programmers must write modules of test code for approximately 1,000 functions, each function having on an average of 3-4 parameters." and 52-55, "The test utility coordinates the parametric testing of each function in the functional programming interface. The test utility reads the script file and generates a plurality of unique test cases.").*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Corrie into the teaching of Noden to include repeating the performing and the revising for each core scenario of the plurality of core scenarios. The modification would be obvious because one of ordinary skill in the art would be motivated to save money by preventing expensive product updates to correct errors, and the high

product quality ensured by the thorough testing would enhance a software product's reputation and marketability *(see Corrie – Column 2: 22-26)*.

As per **Claim 41**, the rejection of **Claim 40** is incorporated; and Noden further discloses:

- producing an aggregate component for each core scenario of the plurality of core scenarios *(see Column 4: 14-17, "An API created and executed, according to the method disclosed herein, can be designed at a high level to provide a complete aggregate function that is self-contained (a `stateless` API).")*.

As per **Claim 42**, the rejection of **Claim 34** is incorporated; and Noden further discloses:

- producing an aggregate component that has a respective relationship with each respective factored type of a plurality of factored types *(see Column 4: 14-17, "An API created and executed, according to the method disclosed herein, can be designed at a high level to provide a complete aggregate function that is self-contained (a `stateless` API)."; Column 7: 48-52, "The macro may then be executed against the internal API using a stub. A stub is a small program routine that substitutes for a longer program. The API stub represents the hook that links the macro to the external program.")*.

As per **Claim 43**, the rejection of **Claim 42** is incorporated; and Noden further discloses:

- producing the aggregate component to support the scenario for which the at least one code sample is written *(see Column 4: 14-17, "An API created and executed, according to the*

method disclosed herein, can be designed at a high level to provide a complete aggregate function that is self-contained (a `stateless` API).").

As per **Claim 44**, the rejection of **Claim 42** is incorporated; and Noden further discloses:

- producing the aggregate component to have an exposed relationship with at least one factored type of the plurality of factored types and an encapsulated relationship with at least one other factored type of the plurality of factored types *(see Column 4: 14-17, "An API created and executed, according to the method disclosed herein, can be designed at a high level to provide a complete aggregate function that is self-contained (a `stateless` API)."; Column 7: 48-52, "The macro may then be executed against the internal API using a stub. A stub is a small program routine that substitutes for a longer program. The API stub represents the hook that links the macro to the external program.").*

As per **Claim 45**, the rejection of **Claim 44** is incorporated; and Noden further discloses:

- wherein the at least one other factored type of the plurality of factored types that has the encapsulated relationship with the aggregate component can be handed off by the aggregate component for direct interaction with another component type *(see Column 4: 14-17, "An API created and executed, according to the method disclosed herein, can be designed at a high level to provide a complete aggregate function that is self-contained (a `stateless` API)."; Column 7: 48-52, "The macro may then be executed against the internal API using a stub. A stub is a small program routine that substitutes for a longer program. The API stub represents the hook that links the macro to the external program.").*

As per **Claim 46**, the rejection of **Claim 42** is incorporated; and Noden further discloses:

- wherein the plurality of factored types are designed using an object-oriented methodology *(see Column 6: 44-46, "As part of this process various stubs and copy-books are created; these could include RPG stubs, C header files, VB and Java calling routines.").*

19. **Claim 35** is rejected under 35 U.S.C. 103(a) as being unpatentable over **Noden** in view of **Corrie** as applied to Claim 34 above, and further in view of **Burger**.

As per **Claim 35**, the rejection of **Claim 34** is incorporated; and Noden further discloses:

- deriving the API from the scenario *(see Column 5: 59-67 to Column 6: 1-4, "Once the desired macros are created, the external customer may utilize the macros in client application(s) and thereby take advantage of the functionality offered by the target application(s). The dispatcher fields new API requests from the network (i.e., via a TCP/IP socket) and may perform some initial validation on the message and extracts environment information from the message to determine which internal application instance this particular API request is intended for ... The message is then queued for execution by a free API engine that may be configured for that particular internal application instance."; Column 6: 61-67, "The macro language is key to how an external API is translated into functions that apply to a particular application. Through a series of instructions comprising one or more macro constructs utilizing internal API calls, SQL statements, and conventional language constructs*

(i.e., string processing features), the external API macro may be executed against the internal/closed application.").

However, Noden and Corrie do not disclose:

- writing a plurality of code samples with regard to the scenario, each respective code sample of the plurality of code samples corresponding to a respective programming language of a plurality of programming languages.

Burger discloses:

- writing a plurality of code samples with regard to the scenario, each respective code sample of the plurality of code samples corresponding to a respective programming language of a plurality of programming languages *(see Column 5: 30-36, "In the present invention, however, a plurality of generic APIs 14 are provided to these pre-existing entry points of the DBM APIs 12 and operating system APIs 20 whereby the invention generalizes existing entries to a plurality of applications 16 written in any of a number of predetermined languages.").*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Burger into the teaching of Noden to include writing a plurality of code samples with regard to the scenario, each respective code sample of the plurality of code samples corresponding to a respective programming language of a plurality of programming languages. The modification would be obvious because one of ordinary skill in the art would be motivated to reduce development effort, overall maintenance and support involved in providing external entry points to software products from applications written in a wide variety of languages having different interfacing requirements and specifications *(see Burger – Column 3: 19-37).*

20. **Claim 36** is rejected under 35 U.S.C. 103(a) as being unpatentable over **Noden** in view of **Corrie** as applied to Claim 34 above, and further in view of **Hayes**.

As per **Claim 36**, the rejection of **Claim 34** is incorporated; however, Noden and Corrie do not disclose:

- determining, by an API designer, if the derived API is more complex than desired; if the derived API is determined to be more complex than desired, then refining, by the API designer, the derived API to produce a refined API; and determining, by the API designer, if the refined API is more complex than desired.

Hayes discloses:

- determining, by an API designer, if the derived API is more complex than desired; if the derived API is determined to be more complex than desired, then refining, by the API designer, the derived API to produce a refined API; and determining, by the API designer, if the refined API is more complex than desired *(see Column 1: 40-42, "Plug-in API's have become more complex as the functionality provided to client applications through plug-in modules has become more sophisticated." and 66-67 through Column 2: 1-3, "It is therefore an object of the invention to provide a host framework which allows a client application to utilize a single simple interface in order to access plug-in modules conforming to any of several past and future native plug-in API's.").*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Hayes into the teaching of Noden to include

determining, by an API designer, if the derived API is more complex than desired; if the derived API is determined to be more complex than desired, then refining, by the API designer, the derived API to produce a refined API; and determining, by the API designer, if the refined API is more complex than desired. The modification would be obvious because one of ordinary skill in the art would be motivated to provide a simple API *(see Hayes – Column 1: 60-63)*.

21. **Claims 49 and 52-59** are rejected under 35 U.S.C. 103(a) as being unpatentable over **Noden** in view of **Hayes**.

As per **Claim 49**, Noden discloses:

- deriving at least one aggregate component to support at least one code sample for at least one scenario *(see Column 5: 59-67 to Column 6: 1-4, "Once the desired macros are created, the external customer may utilize the macros in client application(s) and thereby take advantage of the functionality offered by the target application(s). The dispatcher fields new API requests from the network (i.e., via a TCP/IP socket) and may perform some initial validation on the message and extracts environment information from the message to determine which internal application instance this particular API request is intended for ... The message is then queued for execution by a free API engine that may be configured for that particular internal application instance."; Column 6: 61-67, "The macro language is key to how an external API is translated into functions that apply to a particular application. Through a series of instructions comprising one or more macro constructs utilizing internal API calls, SQL statements, and conventional*

language constructs (i.e., string processing features), the external API macro may be executed against the internal/closed application.");

- determining additional requirements with respect to the at least one scenario *(see Column 6: 49-54, "A complete API definition for a given application may provide all information needed by a separate application to interface to it, i.e., how initiation/termination are performed, what interface protocol is used, what information is sent/received, timing requirements, and other attributes.")*;

- if it is decided that the additional requirements can not be added to the at least one aggregate component without adding more complexity than desired to the at least one scenario, then:

- defining a plurality of factored types responsive to the deciding *(see Column 4: 14-17, "An API created and executed, according to the method disclosed herein, can be designed at a high level to provide a complete aggregate function that is self-contained (a `stateless` API)."; Column 7: 48-52, "The macro may then be executed against the internal API using a stub. A stub is a small program routine that substitutes for a longer program. The API stub represents the hook that links the macro to the external program.")*;

- realizing the at least one aggregate component in one or more processor-accessible storage media *(see Column 16: 2-5, "A system for developing intermediate application programming interfaces comprising a processor ...")*; and

- realizing the plurality of factored types in the one or more processor-accessible storage media *(see Column 16: 2-5, "A system for developing intermediate application programming interfaces comprising a processor ...")*; and

- if it is decided that the additional requirements can be added to the at least one aggregate component without adding more complexity than desired to the at least one scenario, then:

- realizing the refined at least one aggregate component in the one or more processor-accessible storage media *(see Column 16: 2-5, "A system for developing intermediate application programming interfaces comprising a processor ...")*.

However, Noden does not disclose:

- deciding if the additional requirements can be added to the at least one aggregate component without adding more complexity than desired to the at least one scenario; and

- refining the at least one aggregate component to incorporate the additional requirements.

Hayes discloses:

- deciding if the additional requirements can be added to the at least one aggregate component without adding more complexity than desired to the at least one scenario *(see Column 1: 35-37, "... the Photoshop native plug-in API is defined by Adobe developers." and 40-42, "Plug-in API's have become more complex as the functionality provided to client applications through plug-in modules has become more sophisticated.")*; and

- refining the at least one aggregate component to incorporate the additional requirements *(see Column 1: 66-67 through Column 2: 1-3, "It is therefore an object of the invention to provide a host framework which allows a client application to utilize a single simple interface in order to access plug-in modules conforming to any of several past and future native plug-in API's.")*.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Hayes into the teaching of Noden to include deciding if the additional requirements can be added to the at least one aggregate component without adding more complexity than desired to the at least one scenario; and refining the at least one aggregate component to incorporate the additional requirements. The modification would be obvious because one of ordinary skill in the art would be motivated to provide a simple API *(see Hayes – Column 1: 60-63).*

As per **Claim 52**, the rejection of **Claim 49** is incorporated; and Noden further discloses:

- deriving the at least one aggregate component with methods, defaults, and abstractions to support the at least one code sample for the at least one scenario *(see Column 4: 14-17, "An API created and executed, according to the method disclosed herein, can be designed at a high level to provide a complete aggregate function that is self-contained (a `stateless` API)."; Column 7: 48-52, "The macro may then be executed against the internal API using a stub. A stub is a small program routine that substitutes for a longer program. The API stub represents the hook that links the macro to the external program.").*

As per **Claim 53**, the rejection of **Claim 49** is incorporated; however, Noden does not disclose:

- refining the at least one code sample according to the at least one derived aggregate component;
- evaluating the refined at least one code sample with regard to simplicity; and

- repeating the deriving if the refined at least one code sample fails to be as simple as desired as determined in the evaluating.

Hayes discloses

- refining the at least one code sample according to the at least one derived aggregate component *(see Column 1: 66-67 through Column 2: 1-3, "It is therefore an object of the invention to provide a host framework which allows a client application to utilize a single simple interface in order to access plug-in modules conforming to any of several past and future native plug-in API's.")*;

- evaluating the refined at least one code sample with regard to simplicity *(see Column 1: 35-37, "... the Photoshop native plug-in API is defined by Adobe developers." and 40-42, "Plug-in API's have become more complex as the functionality provided to client applications through plug-in modules has become more sophisticated.")*; and

- repeating the deriving if the refined at least one code sample fails to be as simple as desired as determined in the evaluating *(see Column 1: 35-37, "... the Photoshop native plug-in API is defined by Adobe developers." and 40-42, "Plug-in API's have become more complex as the functionality provided to client applications through plug-in modules has become more sophisticated.")*.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Hayes into the teaching of Noden to include refining the at least one code sample according to the at least one derived aggregate component; evaluating the refined at least one code sample with regard to simplicity; and repeating the deriving if the refined at least one code sample fails to be as simple as desired as determined in

the evaluating. The modification would be obvious because one of ordinary skill in the art would be motivated to provide a simple API *(see Hayes – Column 1: 60-63).*

As per **Claim 54**, the rejection of **Claim 49** is incorporated; and Noden further discloses:

- determining additional requirements with respect to the at least one scenario, wherein the additional requirements include additional scenarios, additional usages, and additional interactions with other component types *(see Column 6: 49-54, "A complete API definition for a given application may provide all information needed by a separate application to interface to it, i.e., how initiation/termination are performed, what interface protocol is used, what information is sent/received, timing requirements, and other attributes.").*

As per **Claim 55**, the rejection of **Claim 49** is incorporated; and Noden further discloses:

- considering whether adding the additional requirements to the at least one aggregate component hinders a create-set-call usage pattern *(see Column 5: 59-67 to Column 6: 1-4, "Once the desired macros are created, the external customer may utilize the macros in client application(s) and thereby take advantage of the functionality offered by the target application(s). The dispatcher fields new API requests from the network (i.e., via a TCP/IP socket) and may perform some initial validation on the message and extracts environment information from the message to determine which internal application instance this particular API request is intended for ... The message is then queued for execution by a free API engine that may be configured for that particular internal application instance."; Column 6: 61-67, "The macro language is key to how an external API is translated into functions that apply to a*

particular application. Through a series of instructions comprising one or more macro constructs utilizing internal API calls, SQL statements, and conventional language constructs (i.e., string processing features), the external API macro may be executed against the internal/closed application.").

As per **Claim 56**, the rejection of **Claim 49** is incorporated; and Noden further discloses:

- defining the plurality of factored types responsive to the deciding with a factoring of a full set of functionality *(see Column 4: 14-17, "An API created and executed, according to the method disclosed herein, can be designed at a high level to provide a complete aggregate function that is self-contained (a `stateless` API)."; Column 7: 48-52, "The macro may then be executed against the internal API using a stub. A stub is a small program routine that substitutes for a longer program. The API stub represents the hook that links the macro to the external program.").*

As per **Claim 57**, the rejection of **Claim 49** is incorporated; and Noden further discloses:

- defining the plurality of factored types responsive to the deciding using one or more object-oriented methodologies *(see Column 6: 44-46, "As part of this process various stubs and copy-books are created; these could include RPG stubs, C header files, VB and Java calling routines.").*

As per **Claim 58**, the rejection of **Claim 49** is incorporated; and Noden further discloses:

- determining whether the at least one aggregate component is to encapsulate or expose the functionality of each factored type of the plurality of factored types *(see Column 4: 14-17, "An API created and executed, according to the method disclosed herein, can be designed at a high level to provide a complete aggregate function that is self-contained (a `stateless` API)."; Column 7: 48-52, "The macro may then be executed against the internal API using a stub. A stub is a small program routine that substitutes for a longer program. The API stub represents the hook that links the macro to the external program.").*

As per **Claim 59**, the rejection of **Claim 49** is incorporated; however, Noden does not disclose:

- refining the plurality of factored types to support the at least one aggregate component and the additional requirements.

Hayes discloses:

- refining the plurality of factored types to support the at least one aggregate component and the additional requirements *(see Column 1: 66-67 through Column 2: 1-3, "It is therefore an object of the invention to provide a host framework which allows a client application to utilize a single simple interface in order to access plug-in modules conforming to any of several past and future native plug-in API's.").*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Hayes into the teaching of Noden to include refining the plurality of factored types to support the at least one aggregate component and the

additional requirements. The modification would be obvious because one of ordinary skill in the art would be motivated to provide a simple API *(see Hayes – Column 1: 60-63).*

22. **Claim 51** is rejected under 35 U.S.C. 103(a) as being unpatentable over **Noden** in view of **Hayes** as applied to Claim 49 above, and further in view of **Burger**.

As per **Claim 51**, the rejection of **Claim 49** is incorporated; and Noden further discloses:

- selecting a plurality of core scenarios for a feature area, the plurality of core scenarios including the at least one scenario *(see Column 7: 59-63, "A particular macro may relate to a particular internal application or class of applications (i.e., billing applications) but the technique and the macro language are generic. Thus it is possible that a macro may be coded to serve multiple applications.")*; and

- wherein the deriving comprises: deriving a plurality of aggregate components, which include the at least one aggregate component, to support the plurality of code samples for the plurality of core scenarios *(see Column 4: 14-17, "An API created and executed, according to the method disclosed herein, can be designed at a high level to provide a complete aggregate function that is self-contained (a `stateless` API)."; Column 7: 48-52, "The macro may then be executed against the internal API using a stub. A stub is a small program routine that substitutes for a longer program. The API stub represents the hook that links the macro to the external program.").*

However, Noden does not disclose:

- writing a plurality of code samples showing preferred lines of code for the plurality of core scenarios, the plurality of code samples including the at least one code sample.

Burger discloses:

- writing a plurality of code samples for the plurality of core scenarios, the plurality of code samples including the at least one code sample *(see Column 5: 30-36, "In the present invention, however, a plurality of generic APIs 14 are provided to these pre-existing entry points of the DBM APIs 12 and operating system APIs 20 whereby the invention generalizes existing entries to a plurality of applications 16 written in any of a number of predetermined languages.")*.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Burger into the teaching of Noden to include writing a plurality of code samples for the plurality of core scenarios, the plurality of code samples including the at least one code sample. The modification would be obvious because one of ordinary skill in the art would be motivated to reduce development effort, overall maintenance and support involved in providing external entry points to software products from applications written in a wide variety of languages having different interfacing requirements and specifications *(see Burger – Column 3: 19-37)*.

Official Notice is taken that it is old and well known within the computing art to show preferred lines of code. Software editing programs commonly show preferred lines of code as a feature of the user interface. Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to show preferred lines of code. The modification would be obvious because one of ordinary skill in the art would be motivated to enhance usability.

Response to Arguments

23. Applicant's arguments with respect to Claims 1, 15, 34, and 47-49 have been considered, but are moot in view of the new ground(s) of rejection.

In the remarks, Applicant argues that:

a) With regard to claims 29, 30, and 31, it is respectfully submitted that the "relative" terminology is internally linked so as to make it definite. One of ordinary skill in the art, in conjunction with the disclosure of the instant Paten (sic) Application, would know what is meant by "level of abstraction" and would understand which of two layers is relatively lower and which is relatively higher. With regard to claim 48, it is respectfully submitted that the allegedly "relative" terminology is internally linked so as to make "simpler", "increasingly complex", "progression", etc. definite within the claim.

Examiner's response:

a) Examiner disagrees. The phrases "relatively higher," "relatively lower," and "relatively greater" are indefinite because the specification lacks some standard for measuring the degree intended. The phrases "simpler situations" and "complex situations" are indefinite because the meaning of a term cannot depend on the unrestrained, subjective opinion of the person practicing the invention. See MPEP § 2173.05(b).

In the remarks, Applicant argues that:

b) Moreover, it is respectfully submitted that Corrie, Jr., et al. is directed to standard parameter testing on functions. It is not related to performing one or more usability studies on the API utilizing a plurality of developers (e.g., from dependent claim 5).

Examiner's response:

b) Examiner disagrees. Corrie clearly discloses performing one or more usability studies on the API utilizing a plurality of developers *(see Column 2: 4-8, "When testing all of the functions in the application programming interface to Microsoft Windows, for example, programmers must write modules of test code for approximately 1,000 functions, each function having on an average of 3-4 parameters.")*.

Furthermore, Applicant's arguments fail to comply with 37 CFR 1.111(b) because they amount to a general allegation that the claims define a patentable invention without specifically pointing out how the language of the claims patentably distinguishes them from the references.

Conclusion

24. The prior art made of record and not relied upon is considered pertinent to Applicant's disclosure.

Any inquiry concerning this communication or earlier communications from the Examiner should be directed to Qing Chen whose telephone number is 571-270-1071. The Examiner can normally be reached on Monday through Thursday from 7:30 AM to 4:00 PM. The Examiner can also be reached on alternate Fridays.

If attempts to reach the Examiner by telephone are unsuccessful, the Examiner's supervisor, Wei Zhen, can be reached on 571-272-3708. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the TC 2100 Group receptionist whose telephone number is 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

MARY STEELMAN
PRIMARY EXAMINER

QC / RC
November 9, 2007